Facultat d'Informàtica de Barcelona (FIB),
Universitat Politècnica de Catalunya (UPC)—BarcelonaTech

Visualization of ensembles of molecular simulation trajectories

Master in Innovation and Research in Informatics

Master's thesis

Author: David Duran Rosich

Supervisor: Pere Pau Vázquez Alcocer
ViRVIG group

Supervisor: Àlvar Vinacua Pla
ViRVIG group

19 April 2018

Acknowledgments

I would like to thank my supervisors, Pere-Pau Vázquez and Àlvar Vinacua, for their support and guidance throughout the realization of this thesis.

I would like to thank our colleagues of University of Masaryk, Drs Barbora Kozlíková and Sérgio M. Marques, who greatly contributed to this work by providing us the data sets and biochemical expertise.

I would like to thank Drs Pedro Hermosilla and Timo Ropinski for our fruitful discussions and the creation of MolecularViewer.

Finally, I would like to thank all the received support from my family, friends and colleagues.

Contents

1 Introduction

Molecular Simulations are a set of techniques that use computers to simulate the physical and chemical behavior of atoms and molecules. These simulations are used by researchers of many areas such as pharmacology or biotechnology. For instance, in the context of this thesis, *i. e.* in pharmacology drug design and enzyme catalyst analysis, MS aim at predicting the binding mode and affinity of a small molecule, the ligand (*i. e.* the drug in pharmacology drug design), with a larger biomolecule, the protein. Such a binding is important as it may inhibit or activate certain function of the biomolecule, such as the transmission of signals that communicate pain, which results in a therapeutic benefit for the patient.

Several modeling techniques exists in MS albeit the most popular ones are Molecular Dynamics (MD) and Monte Carlo based techniques (MC). This work is centered on molecular dynamics simulations, which usually involve numerically solving Newton's equations of motions that depend on, among others, the calculated potential energies of the system such as the van der Waals energy or the electrostatic energy. The resulting data is usually a trajectory composed by a temporal series of uniformly spaced frames (snapshots) that encode all the atom positions of the molecular system at a given time rate (*i. e.* the inverse of the so called time-step).

To produce meaningful results, the MD simulations must have a small time-step in order to avoid discretization artifacts, *i. e.* they must be smaller than the life-span of the shortest event to capture, which might be as short as in the order of femtoseconds (10^{-15} seconds). Additionally, the whole simulation should be long enough relative to the time scale of the process being studied which may span in the order of nanoseconds (10^{-9}) or even microseconds (10^{-6}). This inevitably leads to extremely long trajectories which, unfortunately, still requires a high amount of human analysis on the outcome and discussion of potential modifications to the simulated drug. This is further aggravated by the fact that, to the best of our knowledge, no tools exist that specifically deal with the analysis of very large trajectories, and thus, back and forth chart inspection or zoom-in

and out with traditional tools may become tedious and extremely time consuming. In this work we present a novel system for visual exploration of very large trajectories that will aid the researcher in filtering the most relevant parts in an interactive and user-friendly way while providing an informative 3D view as well as an enhanced plot chart.

1.1 Molecular simulation background

Proteins are large biomolecules composed by sequences of amino acid residues. There are 20 types of amino acids in the genetic code. All of them, share a common structure which can be divided into the atoms forming the backbone and the ones forming the side-chain, which gives the specific physico-chemical properties of each residue. These amino acid residues are connected by peptide bonds (see Figure 1.1) and the order, type and length of these sequences define the characteristics of the protein. The different linear chains of amino acid residues are called polypeptide and are commonly denominated the primary structure of the protein.

The 3D structure of a protein is not the one of multiple straight linear sequences but rather the peptide bonds of the amino acid chains fold due to the formation of hydrogen bonds between non-consecutive residues. These hydrogen bonds create characteristic patterns known as α-helix and β-sheet that conform what is called the secondary structures of the protein (see Figure 1.2). The protein folding can temporally vary (and so their secondary structures) which can be crucial so as to allow the entrance of the ligand (*i. e.* the drug in drug design) to the binding site, or to create the specific conformation that define the binding site.

Finally, the binding affinity, *i. e.* the strength of the binding, between the ligand and the protein is usually approximated by the sum of their interaction energies. A conformation is determined to be stable when the total energy is low, *i. e.* negative with a high absolute value, therefore real bound conformations will probably only occur within those conformations. Typically, the non-covalent or non-bonded energies can be divided into the van der Waals energy (short-range) and the electrostatic energy (long-range). Moreover, when simulating the molecules in a solvent, this must be either accounted explicitly, *i. e.* considering all the solvent molecules and their interaction energies as well as with the protein and the ligand, or implicitly, for instance considering the force field (corresponding to the energies) by using a mean-field approach.

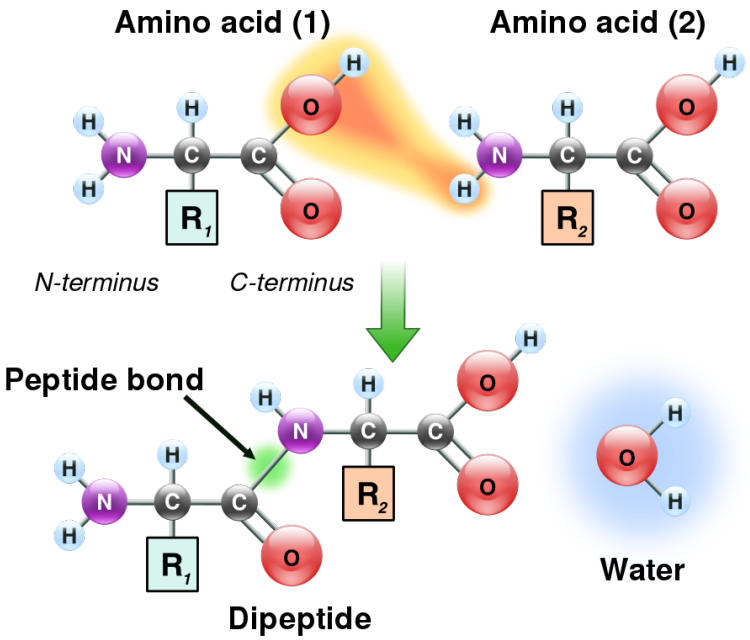


Figure 1.1: Formation of a peptide bond between two amino acids. The resulting chain is called dipeptide (*i. e.* a polypeptide of two elements) and the water molecule is the by-product of the peptide bond. The R boxes constitute the side chain of each amino acid while the other depicted atoms comprise the backbone.

1.2 Our contribution

In this thesis we present an exploration system tailored for very large molecular simulations which may involve several ligands. These conditions impose various challenges as the system must, in general, include:

1. Efficient management of the trajectory data, which may amount to several gigabytes.
2. Presence of suitable informative representations of the whole trajectory that allow for a progressive exploration to fine-grained details. This becomes an even greater challenge when considering several ligands at the same trajectory.
3. Guides that help the researcher in jumping to the relevant trajectory parts, thus mitigating tedious tasks such as extensive back and forth chart inspection.
4. Interaction techniques that facilitate instant and progressive exploration of the data.

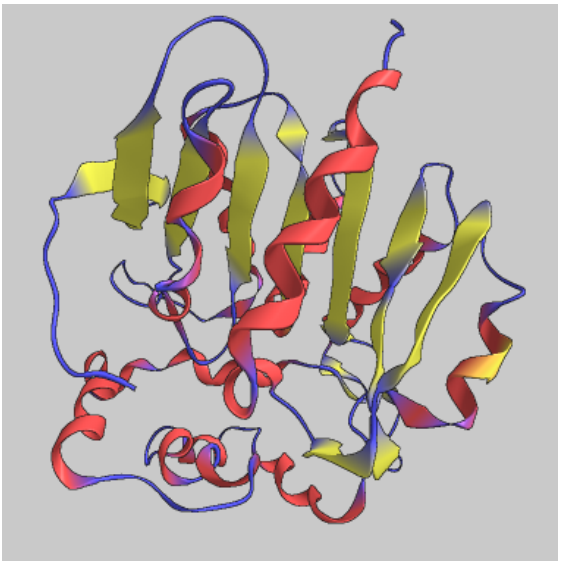


Figure 1.2: Ribbon representation of the dhaA haloalkane dehalogenase protein. The red structures are the α-helices while the yellow structures are the β-sheets. The α-helix pattern is characterized by hydrogen bonds formed between residues separated by approximately four positions in the backbone, whereas the β-sheet pattern corresponds to two distant parts of the backbone interconnected by hydrogen bonds between several of the amino acid residues.

Our system aids the researcher by providing insight at a high abstraction level in order to cope with the inherent complexity of representing huge amounts of data in a meaningful way. In this regard, one key feature is its enhanced chart where several relevant magnitudes are plotted, *e. g.* the total energies of the system or the relative distance between the ligand and the protein. The plot progressively adds details according to the current level of zoom and it contains actionable widgets derived from the regions of potential interest, *i. e.* possible binding zones, that it automatically detects. The user may click on any of those depicted regions to easily jump to these parts of the simulation and examine them in detail. As a result, inspection of extremely long trajectories is greatly facilitated.

Furthermore, we bidirectionally linked a 3D view of a single step of the trajectory with the chart. From the charts to the 3D view the user may jump to any point of the trajectory and update the current depicted frame in the 3D view. Conversely, regions of the 3D view can be selected and the system will highlight on the charts where the selected 3D region is visited by the ligand. This is extremely convenient when examining specific known binding sites or, on the other way, to closely inspect the

3D conformations of the automatically detected regions of potential interest. Finally, each ligand is represented on its own coordinated chart that facilitate the exploration of multiple ligands at once.

Our system have been evaluated by a colleague domain expert, Sérgio M. Marques[1], who found the system of great utility. Hence, due to its attested usefulness and novelty, the presented work have been submitted to the 2018 IEEE Scientific Visualization (SciVis) conference. We provide as supplemental material the submitted video for the conference as it greatly complements this thesis and provides a more realistic depiction and feel of the system as a whole.

The contents of this thesis are organized as follows. First, in Chapter 2, the state-of-the-art for molecular simulation visualization is presented. In Chapter 3 the data we used for testing our application is described. Chapter 4 covers the presented visualization system as well as providing implementation details on major features. Then, results are discussed in Chapter 5. Finally, Chapter 6 concludes this thesis discussing future features for our system.

[1] https://www.muni.cz/en/people/233119-sergio-manuel-marques

2 Previous work

Visualization of biomolecular structures have been in the scope of researchers for decades. Several approaches exist, with different degree of interaction, from atomic level representations to ensembles of biomolecular structures. We refer the reader to the work of Kozlíková et al. [18] for a complete summary of the traditional and novel techniques in biomolecules visualization.

The growing capabilities of modern GPU allowed for the visualization in real-time of large molecules and molecular systems. Several methods have been proposed on the visualization of large biomolecular systems such as the ones by Chavent et al. [5]. Other examples include the MegaMol prototyping framework for real-time visualization of large particle-based scenes by Grottel et al. [12], or cellView published by Le Muzic et al. [21]. However, all these methods only deal with the real-time visualization of large biomolecular systems and do not consider the problem of large biomolecular trajectories exploration.

In the context of this thesis, *i. e.* exploration of molecular simulations in the fields of drug design and enzymatic catalysis studies, trajectories mostly capture the process on which a small molecule (ligand/drug) arrive at the protein active site, where it may bind to the protein receptor or enzyme. The existence and classification of the protein active sites is, therefore, of utmost importance and several approaches for their detection and visualization have been already proposed. A comprehensive overview of the existing algorithms and visualization methods for cavities was recently published by Krone et al. [19]. Nevertheless, this problem is conceptually different from what we are trying to achieve, as we do not concern ourself in the protein active sites identification and classification but rather on the exploration of the trajectory itself.

It is worth mentioning that our work is partially related to the visualization of large data sets, in which the complexity of the data often originates from its variation or volume [9]. In this field, the recent approaches commonly have as requirement the *interactive exploration* of the data. However, all the methods largely differ depending on

the nature of the underlying information and the queries that need to be supported [11]. When visualizing the whole dataset, several approaches rely on the aggregation of data (*e. g.* [6, 34, 35, 36]) and progressive exploration [7, 32, 33] on which the details of the data are presented to the user in a on-demand basis. Commonly, one of the main goals is finding suitable abstract representations, like the work of Liu et al. [25] on which they present several motifs that encode consecutive events that appear frequently together, and derive new informative data, such as patterns [24], in order to shed light upon specific analysis like in the work of Malik et al. [27] with cohort comparison of patient data.

In this work we adapted some of this well founded concepts, such as data aggregation, data abstraction and progressive exploration, to our specific type of data. However, contrary to most of the previously mentioned methods, our data is two-fold: for each step in the time sequence there is both a 3D configuration of atoms and a set of magnitudes such as the energy values or relative distances. Therefore, it is not only necessary to cope with the large amount of information in terms of time steps, but finding appropriate techniques to link both types of data, the 3D conformations and the 2D charts, is of utmost importance since the 3D view helps researchers to fully understand and complement the information that is displayed by the trajectory charts.

More aligned to our work, several approaches to visual exploration of molecular dynamics exist. Lindow [23] presented so called dynamic channels whose visual exploration enables the users to analyze the time evolution of the cavity over time. Their proposed solution integrates several visualization methods, spanning from static overview representations to animations of trajectories. Byška et al. introduced a method for visual exploration of protein tunnels and the surrounding amino acids over time [3] and a specialized visualization technique for detailed exploration of tunnel bottlenecks and their evolution over time [2]. Nevertheless, none of these approaches does not consider trajectories containing the ligand movements as well.

Exploration of trajectories, taking into account the ligand interactions with protein, has been published by Hermosilla et al. [14]. Their tool enables the users to interactively trace the interactions between protein and ligand. Furmanová et al. [10] introduced a system for visual analysis of ligand behavior in large trajectories. It consists of a set of representations enabling the users to identify interesting parts of trajectories, based on user-defined properties. In this work, we employed the same 3D rendering back-end as Hermosilla et al. [14]. The interested reader may find a more comprehensive description of this 3D rendering back-end in the PhD thesis of Hermosilla [13]. However,

as noted in subsection 4.2.1, all the previous works based on [13] only dealt with relatively short trajectories and no exploration mechanism for large trajectories were presented.

Finally, there are several widely-used packages in the biochemical field. For instance, LigPlot+ [20] analyze multiple ligand-protein interactions using a set of 2D planar maps of the 3D configuration. Additionally, it may output a 3D configuration that can be viewed using other programs such as Pymol. However, it does not provide specific tools for progressive simulation explorations for large trajectories. Schrödinger's SID [26] generates a set of static charts that can be written in PNG or SVG formats, but no interactive exploration is provided, nor 3D exploration of multiple snapshots. VMD [16], another popular program, is mainly devoted to the visualization of large molecular complexes, and the analysis of MD trajectories. Nevertheless, it does not contain the 2D to 3D and 3D to 2D linkings we provide, greatly undermining efficient exploration of specific active sites and the 3D conformations that give raise to the different values of the visualized magnitudes. Similar to VMD, PLIP [29] is focused on the analysis of Protein-ligand interactions, unfortunately, it does not deal with precomputed trajectories, and does not allow the exploration of the MD results. It works as a web service that can read entries from the Protein Data Bank. PyMol [30] is an open source package distributed and maintained by Scrhödinger whose objective is to render and animate 3D structures, not the analysis of MD trajectories. TAMD proposes a similar dashboard view, but its widgets and interaction tools are somewhat limited [22]. Other packages perform the analysis of the trajectories only by extracting information from the simulation, not by providing a unified system for 2D and 3D analysis [28].

3 Input data

This project has been carried out in collaboration of scientists of University of Masaryk, Brno, Czech Republic. Besides some guidance, they have also provided the simulation data. More concretely, for the tests, we used molecular dynamics trajectories of the dhaA haloalkane dehalogenase protein with different ligand configurations provided by biochemists. Table 3.1 provides the summary and Figure 3.1 the 3D rendering of the four different datasets we explored. All the provided files were in AMBER tools format, detailed in section 3.1. The case with three "1,2,3-Trichloropropane" (TCP) impose an inherent challenge to design an extensible solution that works for multiple ligands. Moreover, while some of the datasets could be handled in memory, the solution had to be out-of-core in order to handle the cases that occupy 41 GB and 45 GB of memory, *i. e.* the DCP trajectory of 100 K steps with the solvent and the trajectory of 800 K steps also with the DCP ligand but without the solvent molecules, respectively. Besides the trajectories, the biochemists provided us with the electrostatic and van der Waals energies for the 3 TCP dataset.

Ligand type	CF	# ligands	Solvent?	# steps	Storage size
2,3-Dichloro-1-propanol (DCP)	$C_3H_6Cl_2O$	1	yes	50 K	20.8 GB
2,3-Dichloro-1-propanol (DCP)	$C_3H_6Cl_2O$	1	yes	100 K	41.6 GB
2,3-Dichloro-1-propanol (DCP)	$C_3H_6Cl_2O$	1	no	800 K	45.0 GB
1,2,3-Trichloropropane (TCP)	$C_3H_5Cl_3$	3	no	50 K	2.8 GB

Table 3.1: The four different MD trajectories we explored. All four of them have the dhaA haloalkane dehalogenase protein (with 293 residues and 4650 atoms) as the main molecule. The first two datasets also include ∼10 K water molecules of the solvent (*i. e.* 30 K atoms).

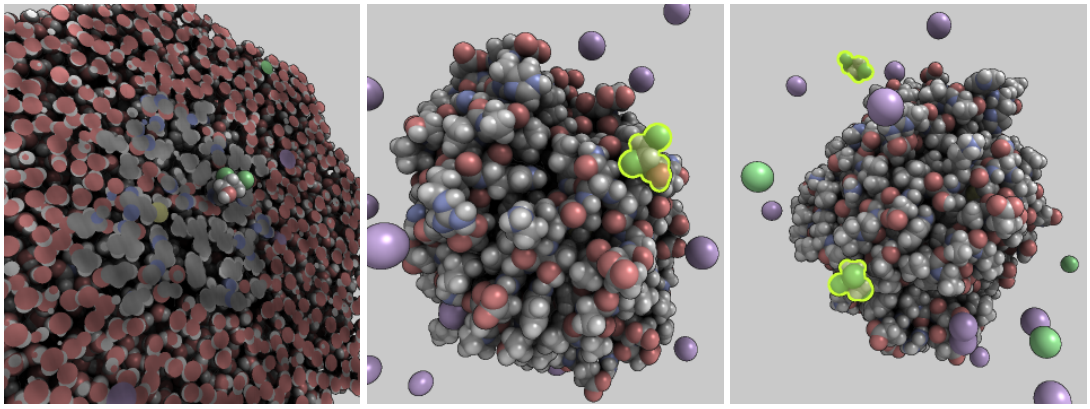


Figure 3.1: 3D renders of the dhaA haloalkane dehalogenase protein with the DCP and the solvent, without the solvent, and with 3 TCP.

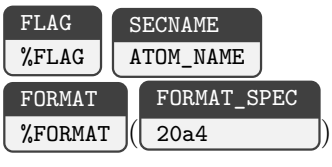
3.1 AMBER trajectory format

`AMBER Tools` [4] usually store molecular trajectory data in multiple files. In our datasets trajectories were split into one parameter/topology file (`.prmtop`) and a NetCDF binary file containing all the atom positions per frame (`.nc`).

The parameter/topology file (`.prmtop`) contains textual data organized in multiple sections in a way that is designed to be easily parsed by Fortran code. Each section is of the form:

```
%FLAG <SECTION NAME>
%COMMENT ...
%FORMAT(<FORTRAN FORMAT>)
... data according to <FORTRAN FORMAT>
```

We created a simple lexer and grammar for the `ANTLR4` software[1] that would identify all such sections with the following tokens:



[1] https://github.com/antlr/antlr4

```
DATA_LINE
N   H1  H2  H3  CA  HA  CB  HB  CG2 HG21HG22HG23CG1 HG12HG13CD1 HD11HD12HD13C
DATA_LINE
O   N   H   CA  HA2 HA3 C   O   N   H   CA  HA  CB  HB  CG2 HG21HG22HG23OG1 HG1
...
```

We ignore all the lines that start with `%COMMENT` (or `%VERSION`). Once all these tokens are identified, using the listener paradigm the actual data is retrieved and stored in internal structures.

For the NetCDF binary file (`.nc`) we use the `libnetcdf-cxx4` library, which is a `C++` wrapper for the `C`'s `libnetcdf` library.

3.2 Derived data

In order to provide visualization and inspection aids of such large trajectories, two magnitudes are computed from the original data: the ligand's speed and the ligand's relative distance to the molecule. Albeit our system is general enough that any 2D temporal measure may be used, those two provide an excellent starting point as they are clear indicators of possible interactions between the molecule and the ligand (*i. e.* small distance) and whether such interactions are stable (*i. e.* low speed).

The datasets we explored dealt with the boundary conditions of the simulation by means of wrapping. Consequently, if the ligand were to leave the simulation space, it is, instead, automatically placed to the other side of the simulation space. This has the undesired effect that the computed velocity is contaminated by outliers orders of magnitude higher than the expected value. Thus, we filtered the velocity values by using the upper Tukey's fence [15] so that values greater than $Q_3 + 3(Q_3 - Q_1)$ were converted to *NaN*, where Q_1 and Q_3 are the first and third quartile, respectively. Except when depicting the value details, all *NaN* values are treated as non-existent and visualized as the value 0. For all the datasets, the number of filtered velocity values represent less than a 1.62 % of the total amount.

We further enhance the data by heuristically analyzing the atoms that each ligand is interacting with at each time step. This heuristic considers the atoms with a distance between the centroid of the ligand and the center of the atom lower than a threshold

(8 Å), but could be easily extended by also considering interacting forces and energies, if such information was provided. For an efficient query of this data we exploited its temporal cohesion (*i. e.* one atom interacting with a ligand at frame f is likely to be interacting at frame $f + 1$) and stored for each atom of the molecule a list of sorted disjoint intervals indicating when the atom is interacting with a particular ligand. For example, if an atom is interacting with a ligand at frames from 4 to 10 and from 31 to 38, the application would store the list $[4, 10], [31, 38]$ for this atom. This information will be thoroughly used for some of the techniques explained in chapter 4.

In order to provide insight on the relevant parts of the trajectory (*i. e.* those where the ligand is interacting with the molecule) we classify per ligand each frame in whether the ligand is *outside* the molecule or, otherwise, interacting with the *surface* or the *interior* of the molecule. This classification is heuristically built using the previously computed list of ligand-atom interaction intervals by considering the list L of atoms that are interacting with the molecule at a particular frame f. Then, frame f is classified according to:

$$\text{if } \begin{cases} |L| \leq 5, & \text{ligand is } \textit{outside}, \\ 5 < |L| \leq 10, & \text{ligand is on the } \textit{surface}, \\ 10 < |L|, & \text{ligand is on the } \textit{surface} \text{ or the } \textit{interior}. \end{cases}$$

In the third case a further test is made in order to distinguish whether the ligand is interacting at the surface or the interior. This test consists in computing a plane that goes through the centroid of the ligand (C) and evaluating the distribution of the atoms of L across the two semispaces. If such distribution is close to be 50-50 then chances are that the ligand is surrounded by the molecule. Specifically, the distribution must be at most 70-30 for the ligand to be considered inside the molecule. From all the infinite planes that go through C, we consider the one oriented towards the average of normalized directions between the atoms in L and C, *i. e.* $N = \sum_{a \in L} \textit{normalized}(a_{xyz} - C_{xyz})$, as this is locally "facing towards the molecule" and, as such, will likely have high disparity when the ligand is not actually surrounded by the molecule (see Figure 3.2).

All this data is pre-computed prior to the data-exploration step as it must be readily available in order to maintain a real-time framerate. Moreover, the bounding box of the whole simulation and the secondary structures of the molecule at each step (computed using the DSSP [17] algorithm provided by the `CppTraj` [28] software from

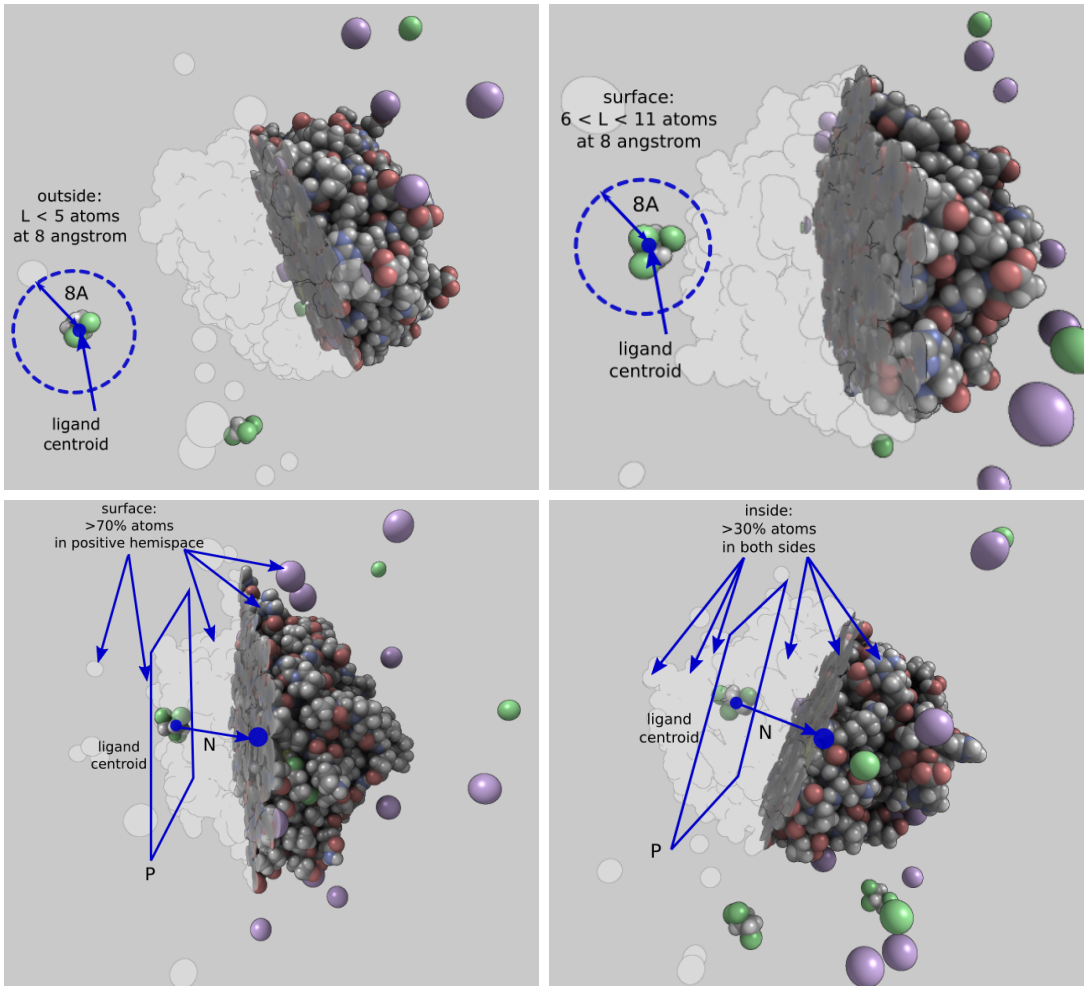


Figure 3.2: The four possible cases of interaction classification. From left to right and top to bottom, the frame is classified as *outside*, *surface*, *surface* and *interior*.

`AMBER Tools`) are also pre-computed as they are required by the 3D renderer library we use.

4 Software

We designed our system for visualizing and exploring large molecular dynamics trajectories. Thus, the system had to provide an appropriate abstraction in which the user could quickly acquire high level insight about the whole simulation without the need of a frame-per-frame exploration, which would soon become tedious and time-consuming as, for instance, completely watching a trajectory of 800 K frames reproduced at a rate of 60 frames per second would require about 3 h and 40 min.

Typically, even for small trajectories, researchers use the energy values as an indicator of how the simulation fares. The reason is that when just considering the energy values, important inquiries can already be answered such as whether the ligand is in a binding position (*i. e.* low energy value) or has been repelled while interacting with the protein (*i. e.* an spike in the energy value). Note that considering just a single value is, of course, a much more abstract and simple representation than the actual information of a trajectory frame which is the whole 3D atom configuration of the molecular system. Altogether, 2D charts fit the role of an abstraction visualization that is both widely used among the researchers community, thus exhibiting a low learning-curve, and informative because, as already mentioned, crucial high-level insight can be acquired at a glance. For this reason, as explained in section 4.3, one of the key pieces of our visualization system are a series of enhanced charts that allow for a progressive exploration of several magnitudes.

While the plotted magnitudes provide a high-level knowledge of the molecular simulation, this is not enough. In order to understand the molecular process the details of the atom configuration are fundamental. To this end, we adapted the `MolecularViewer` library [13] to our needs (see subsection 4.2.1) and incorporated it in order to visualize the 3D conformations of the molecular systems. More details are given in section 4.2.

Finally, we integrated both the 3D view and the 2D charts in a single application (see Figure 4.1 for a general overview). This decision is supported by the fact that, during the

exploratory step, it is frequent that the researchers go back and forth between the energy plots, where potential regions of interest are identified, and the visualization of the 3D conformation of a single frame, in which the detected regions of interest can be thoroughly examined. Thus, integrating in a single application both views greatly facilitates their work-flow provided that we implemented the necessary interaction techniques to jump from the 2D representation to the 3D view, and vice versa.

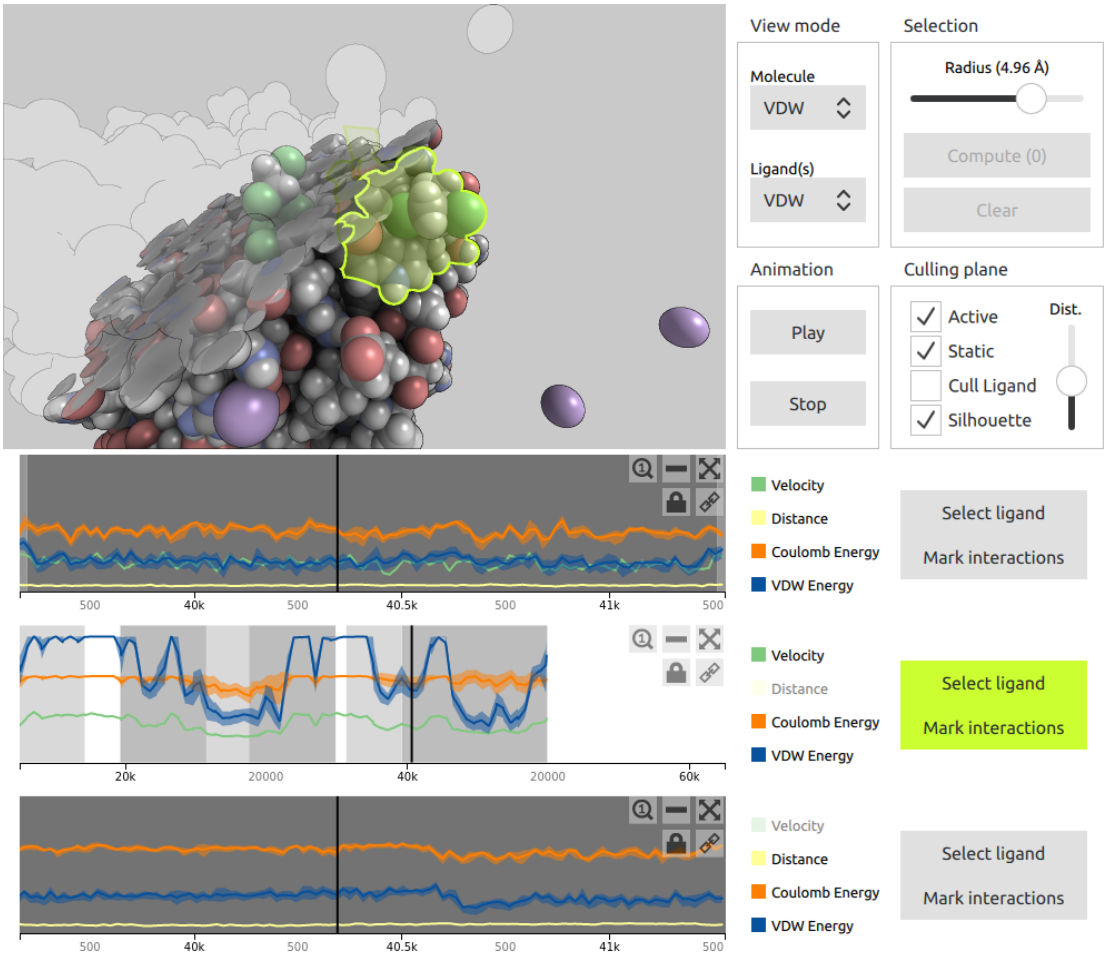


Figure 4.1: General overview of our visualization system.

4.1 Qt-based interface

The application interface is implemented using the Qt 5.10 framework[1]. We used the Qt QML module which provides the QML language, a JavaScript-based language used to specify the UI elements and their behavior. For most of the UI controls (referred as *widgets* in different Qt modules) we used several of the ones provided by the Qt Quick Controls 2. However, the windows for the 3D renderer and the chart viewer had to meet specific requirements and were implemented using the QQuickFramebufferObject and the QtWebEngine, respectively.

Using QQuickFramebufferObject is the new idiomatic way in a Qt-QML application to implement an OpenGL-based UI element. In this setting, the Qt framework is responsible of providing a valid OpenGL context and the OpenGL framebuffer corresponding to the actual screen portion in the user interface. In this paradigm the 3D renderer must be split into the UI element, responsible of reacting to the different received signals originated from user input or other UI elements, and the renderer part, responsible of issuing the actual OpenGL calls, possibly in a different dedicated thread. The UI component enqueues all the received user events which are retrieved by the renderer code at the next synchronize call.

For the interactive chart we used the D3 JavaScript library [1] for producing dynamic and interactive data visualizations in currently available web standards (SVG, HTML5 and CSS). In order to integrate the web-based chart viewer in the Qt-based application we used the QtWebEngine module whose web engine is based on the code of the Chromium project.

4.2 3D view

According to their authors [13], the `MolecularViewer` library is able to load and visualize large molecular models (*e. g.* molecule *31YJ* with 1356 K atoms, see Figure 4.2) at a real-time frame-rate (~ 30 fps). Combined with its wide range of representations and myriad of tools, it made an excellent candidate as the renderer of the molecular 3D conformation of the different frames. Specifically, we highly valued the following characteristics:

[1] https://doc.qt.io/qt-5.10/

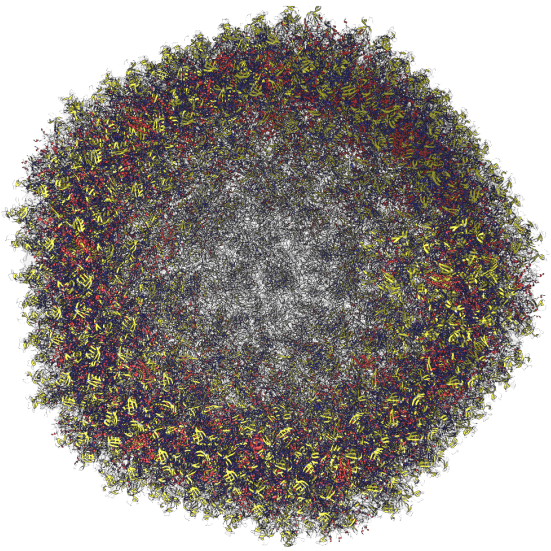


Figure 4.2: Ribbon representation of the molecule (3IYN), with 749 K residues and 5975 K atoms. Image extracted from [13].

- The different powerful representations or motifs of the protein and the ligands, specially the solvent excluded surface representation.
- The efficient object-space ambient occlusion for an enhanced depth perception.
- The flexible highlight/selection mechanism which we used for several of our techniques.
- The clipping plane to inspect the interior of the protein combined with the semi-transparent representation silhouette to preserve the overall 3D form of the protein.

In our application we decouple the representation used for the protein and the one used for the ligand(s) in favor of some useful combined representations such as solvent-excluded surface for the protein and space-filling van der Waals representation for the ligand(s). Figure 4.3 depicts all representations and some of the possible combinations, while a comprehensive list is as follows:

- space-filling *van der Waals* representation, *ball-and-sticks* and *licorice* for atomic-level representations.
- *ribbons* representation where the secondary structures are depicted as red helices for α-helices and yellow arrows for the β-sheets, and finally

- *CSurface* (non-incremental rendering) and *PBCSurface* (incremental rendering) for solvent-excluded surface (SES) representation suited for finding cavities and potential docking sites.

Finally, we modified the `MolecularViewer` to fit our specific needs and introduced several new features required for our techniques. As seen in detail in subsection 4.2.1, these modifications include adding support for the AMBER trajectory file format, the addition of a pre-process step in which the derived magnitudes explained in section 3.2 are computed as well as introducing the 3D selection explained in subsection 4.2.3.

4.2.1 Modifications of the MolecularViewer library

The expected input data of the `MolecularViewer` software are molecular trajectories generated from Monte Carlo simulations where given a frame f several configurations are randomly generated and validated as possible successors of frame f. This paradigm uses big time steps as it is prohibitively expensive not to do so and, therefore, it results in relatively short trajectories (in the order of hundreds of frames per trajectory). Moreover, several branching points may appear as more than one configuration may be a possible successor of a particular frame, hence given the initial configuration the simulator outputs several trajectories.

With these characteristics in mind, `MolecularViewer` had to meet several specifications such as multiple trajectories support as well as dynamic loading of frames for one to analyze one possible trajectory while other branches are being explored by the simulator. Albeit it is capable of loading and visualizing large models, the viewer was not suited for several orders of magnitude smaller molecules but with trajectories that contain a large number (*e. g.* hundreds of thousands) of frames. Hence, for `MolecularViewer` to be the backbone of our 3D rendering it had to undergo several algorithmic and structural changes.

The most important conceptual modification has been shifting from a dynamic definition of a trajectory in which frames may be added or deleted on-the-fly to consider simulations as a single huge trajectory with all its frames known and accessible from the beginning. Albeit considering trajectories to be dynamic offers more flexibility, the inherent algorithmic and paradigm complexity clearly outweighed its benefits given the static nature of our data. Therefore, we revisited all the data management of

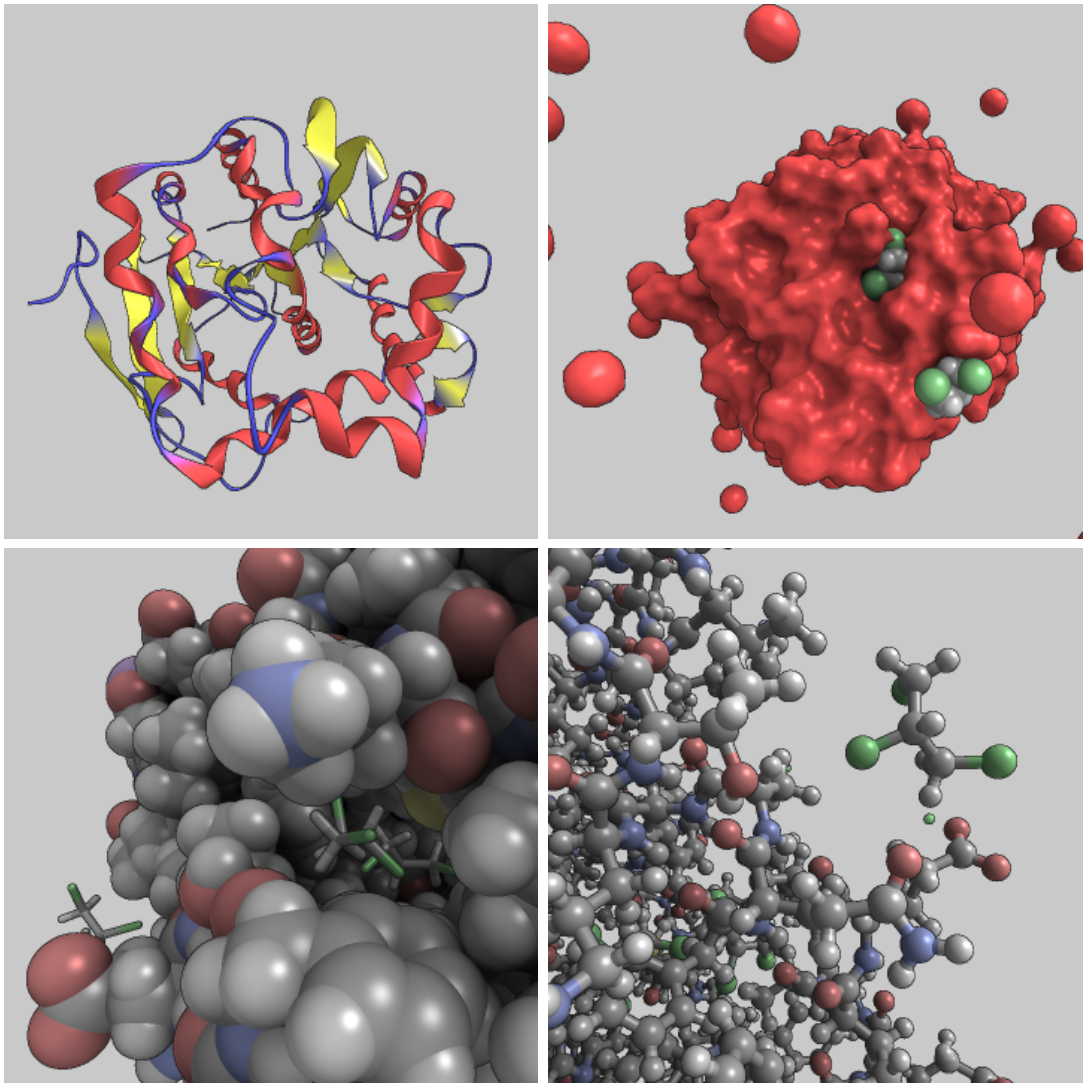


Figure 4.3: Different representations of the DhaA haloalkane dehalogenase with 3 TCP. From left to right, top to bottom, *ribbons*, *CSurface* for the protein combined with *VDW* for the ligands, protein with *VDW* and ligands with *licorice*, *ball-and-sticks* for both the protein and the ligands.

`MolecularViewer` and modified it accordingly. A comprehensive list of such changes is as follows:

- Moved from an out-of-order frame storage with linear random access of an specific frame to a temporally-sorted vector.
- Removed the ability to add/remove frames after initialization.

- Added several pre-computations instead of computing them on demand (*e. g.* the whole trajectory bounding box).
- Dynamically load a single frame at a time (*lazy loading*) while retaining the whole trajectory out-of-core.
- Dynamically send a single frame's data to the GPU instead of the whole trajectory. This required changing how the code for the visual representations handle their data.
- Included support for AMBER file format (see section 3.1).
- Added the computation of derived data (see section 3.2).

One of the biggest changes is having only a single frame at main memory at a time. The reason is that our data does not entirely fit in memory for current workstations at the mid-range specifications making out-of-core a necessity. Our implementation simply loads the whole frame in an on-demand basis and, even though more complex solutions were considered such as dividing the trajectory into chunks and add a predictor that loads on the background the next likely needed chunk, those were not required for our datasets as we already reached a real-time frame-rate.

4.2.2 3D scene interaction

The camera used in the 3D scene can be edited, *i. e.* translated, rotated and zoomed, using mouse drag and wheel.

Several regions of importance lie in the interior of the protein as binding commonly happens in cavities. Like Hermosilla [13], we opted for the use of the clipping plane tool in order to explore the interior of the protein. The usefulness of this tool is two-fold: it allows to identify the interesting residues that interact with the ligand when it is close to a binding position and, combined with a SES representation, it allows the researchers to quickly identify such cavities and regions of interest. Additionally, while inspecting with the clipping plane, the user may retain the silhouette of the clipped parts to preserve the overall volumetric form (see Figure 4.4).

All these interaction techniques were available in the `MolecularViewer` library out-of-the-box and so, no further work was required for implementing them once we integrated the library to our application.

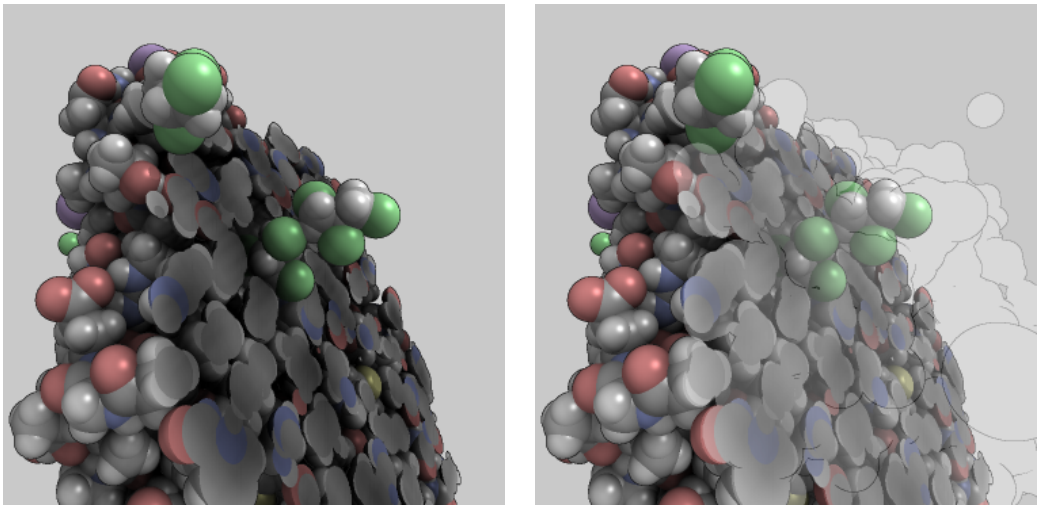


Figure 4.4: The use of the clipping plane allows to inspect the interior of the protein. Additionally, the user may activate the silhouette to preserve the volumetric form.

4.2.3 3D selection

One key element of an integrated application where multiple representations of the data are present is how the different views are linked. While going from an abstract view (*e. g.* the 2D charts) to a detailed representation (*e. g.* the 3D view where the actual molecular system conformation of a single frame is depicted) can be done following the Shneiderman's mantra, where the overview, zoom and filter are performed on the abstract view and the detailed view is updated on demand, the reverse flow is non-trivial. Nevertheless, the 3D view of the molecular system is essential as it provides the researcher crucial information about the characteristics of the protein, more so when combined with the *ribbons* representation to depict the secondary structures of the protein or the solvent excluded surface representation for identifying cavities at a glance. Hence, it was required that the researcher would be able to guide the exploration of the abstract 2D view according to the molecular system conformation visualized in the 3D view.

To that end, we provide a new feature to the `MolecularViewer`: a 3D selection of a spherical region of the protein. The purpose of this selection tool is to allow the researcher to select specific sites of the protein where interaction with the ligand(s) is deemed important. Once selected, the 2D charts are updated accordingly and the intervals in which the ligand(s) interact with the selected atoms are highlighted and their

interaction greatly facilitated. This way, several key inquiries can be quickly answered such as whether any protein residue took an important role in the ligand-protein interactions or when and which ligand(s) interact with a known active site.

Figure 4.5 illustrates the usual workflow of the implemented 3D selection, which is as follows:

1. The user moves the center of the sphere by hovering over the 3D view while pressing the `SHIFT` key. During this step, the actual selected atoms are highlighted (using the `MolecularViewer` built-in highlight mechanism) and updated accordingly so that the user always receive a visual feedback of the currently selected region.

2. Once the user is satisfied with the current position of the sphere, they may fix it with a mouse left click. If, after further inspection, the user considers it is necessary to move the selection sphere they may at any time unfix it by pressing `SHIFT` and mouse right click.

3. Finally, the radius of the selection sphere might be modified with a slider or `SHIFT` plus mouse wheel.

One important consideration in step 1 is how to map the 2D mouse position with the corresponding 3D position. As previously mentioned, the regions of interest usually lie inside the protein, consequently, we designed this tool to interact with the clipping plane in the following way:

- If the clipping plane is active and facing towards the viewer at an angular difference of no more than 60°, the center of the sphere will lie on the plane.

- Otherwise the sphere will lie on the surface of the protein corresponding to the current mouse position.

With this methodology we strove for a balance between accurate selection of arbitrary regions with the help of the clipping plane and convenience for selecting pockets and regions visible from outside the protein.

Once the researcher has confirmed the 3D region (by pressing the `Compute` button), the system is able to derive (as explained in implementation details) at which frame intervals there is a strong interaction between each ligand and the selected region.

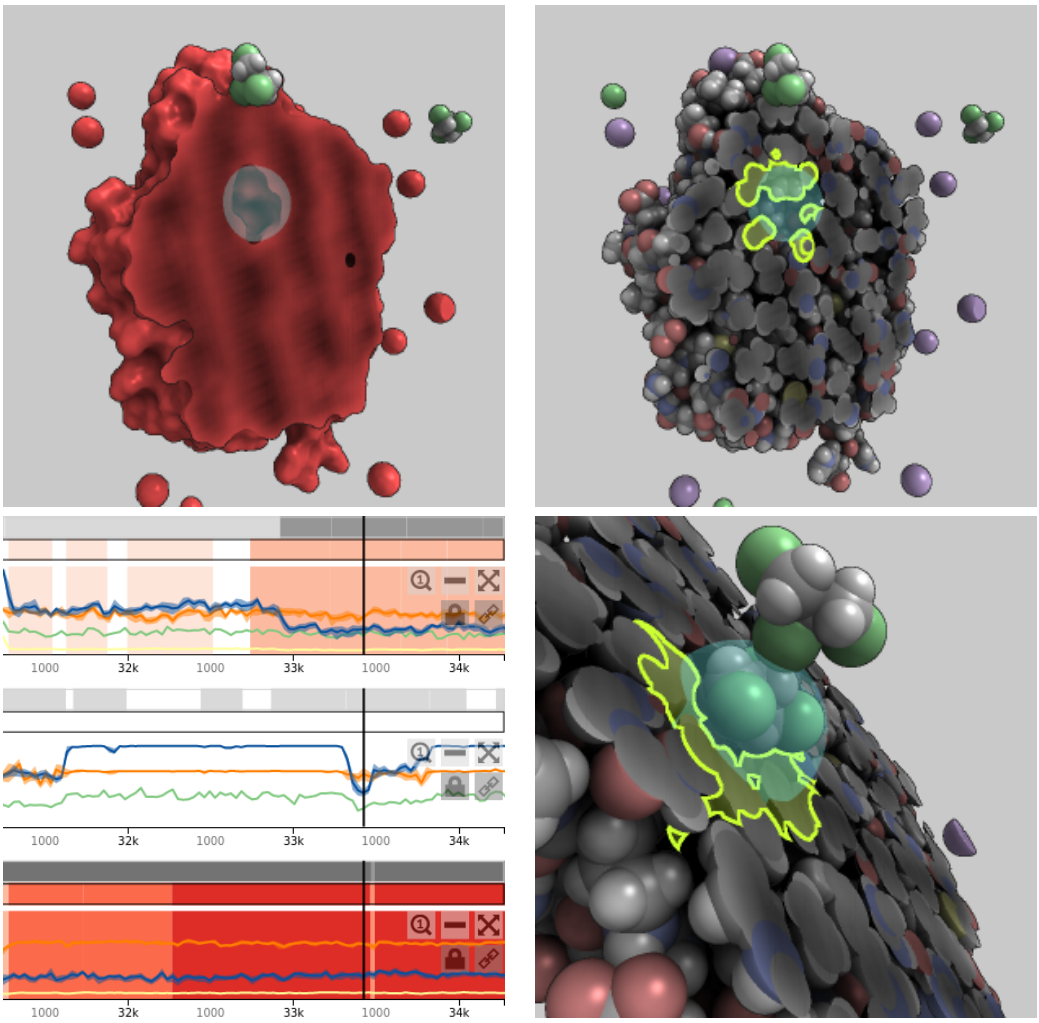


Figure 4.5: Typical use case of our 3D selection tool. First the researcher examines the interior of the protein with a solvent-excluded surface representation in order to find cavities. Once an interesting region has been appropriately selected, the regions of the trajectory in which each ligand interacts with the selection are highlighted. At this point the user can easily navigate in the chart to find interactions with the selected region and update the 3D view accordingly.

Implementation details

The system computes the frame intervals in which the selected atoms interact with the ligand with a 1D line-sweep algorithm. For example, given the following lists of frame intervals in which the atoms interact with ligand l:

$$\begin{aligned} A_1: &\quad [0,4],[20,25] \\ A_2: &\quad [2,8],[20,45] \\ A_3: &\quad [6,30] \\ A_4: &\quad [24,30] \\ A_5: &\quad \emptyset \end{aligned}$$

If the selected atoms are $S = \{A_1, A_3, A_4\}$, the resulting magnitude (list of triplets [<begin>, <end>, w]) is

$$\begin{aligned} R = [0,4,|\{A_1\}| = 1],[6,20,|\{A_3\}| = 1],[20,24,|\{A_1,A_3\}| = 2], \\ [24,25,|\{A_1,A_3,A_4\}| = 3],[25,30,|\{A_3,A_4\}| = 2]. \end{aligned}$$

The time complexity to compute such magnitude is $\mathcal{O}(n \log n)$ where n is the sum of the number of intervals of each selected atom. This magnitude is then normalized dividing it by $|S|$ (which is 3 in the example) and quantized according to the following table:

$w/\|S\|$	Classification
0.0–0.2	*no interaction*
0.2–0.6	*low interaction*
0.6–0.8	*medium interaction*
0.8–1.0	*high interaction*

In the previous example, R would result into

$$\begin{aligned} R' = [0,4,0.33 = \textit{low interaction}],[6,20,0.33 = \textit{low interaction}], \\ [20,24,0.67 = \textit{medium interaction}],[24,25,1 = \textit{high interaction}], \\ [25,30,0.67 = \textit{medium interaction}]. \end{aligned}$$

This magnitude is then hierarchically clustered as explained in subsection 4.3.3.

Finally, besides the 1D line sweep algorithm, the 3D selection method required the render of a sphere whose center was at the corresponding unprojected object-space coordinate of the fragment behind the mouse cursor.

4.2.4 Highlight contact atoms

Visualizing the atoms that interact with the ligand(s) has been a requested feature by the domain expert, as this information is of utmost importance when evaluating the binding configurations. As in the 3D selection, we used as the visual encoding the `MolecularViewer` highlight mechanism. In order to determine which atoms interact with each ligand at a particular frame, the precomputed list of sorted disjoint frame intervals is used, as explained in the implementation details. Albeit this information is heuristically calculated (*i. e.* we consider that an atom is interacting with the ligand if its center is closer than 8 Å) the data did not contain the required information (*i. e.* atom pair-wise energies) for extracting the exact result. It is worth mentioning that the interacting atoms depend on the current 3D conformation of the molecular system and, thus, the list of highlighted atoms must be updated from frame to frame.

Finally, the activation and deactivation of this highlight is done by a couple of buttons next to the each ligand's chart. This way, the link between each ligand and its 3D representation is strengthened as the user may at any time highlight the corresponding 3D ligand of each chart including (or not) the interacting atoms. Figure 4.6 illustrates how this mechanism can be used to identify the corresponding 3D representation of each ligand and how the interaction atoms are depicted.

Implementation details

Using the list of frame intervals in which each atom interact with a particular ligand l, one can know whether an atom interacts with ligand l at frame f using a dichotomic search in $\mathcal{O}(\log k)$ time, with k being the maximum number of intervals any atom has (a fraction no bigger than half the total number of frames in the trajectory, and mostly orders of magnitude smaller). By iterating through all n atoms, the full list of atoms interacting with ligand l at frame f can be retrieved in $\mathcal{O}(n \log k)$ time. This computation is done every time a new frame is visualized as the list of contact atoms varies according to the relative distance between the ligand and each atom of the protein.

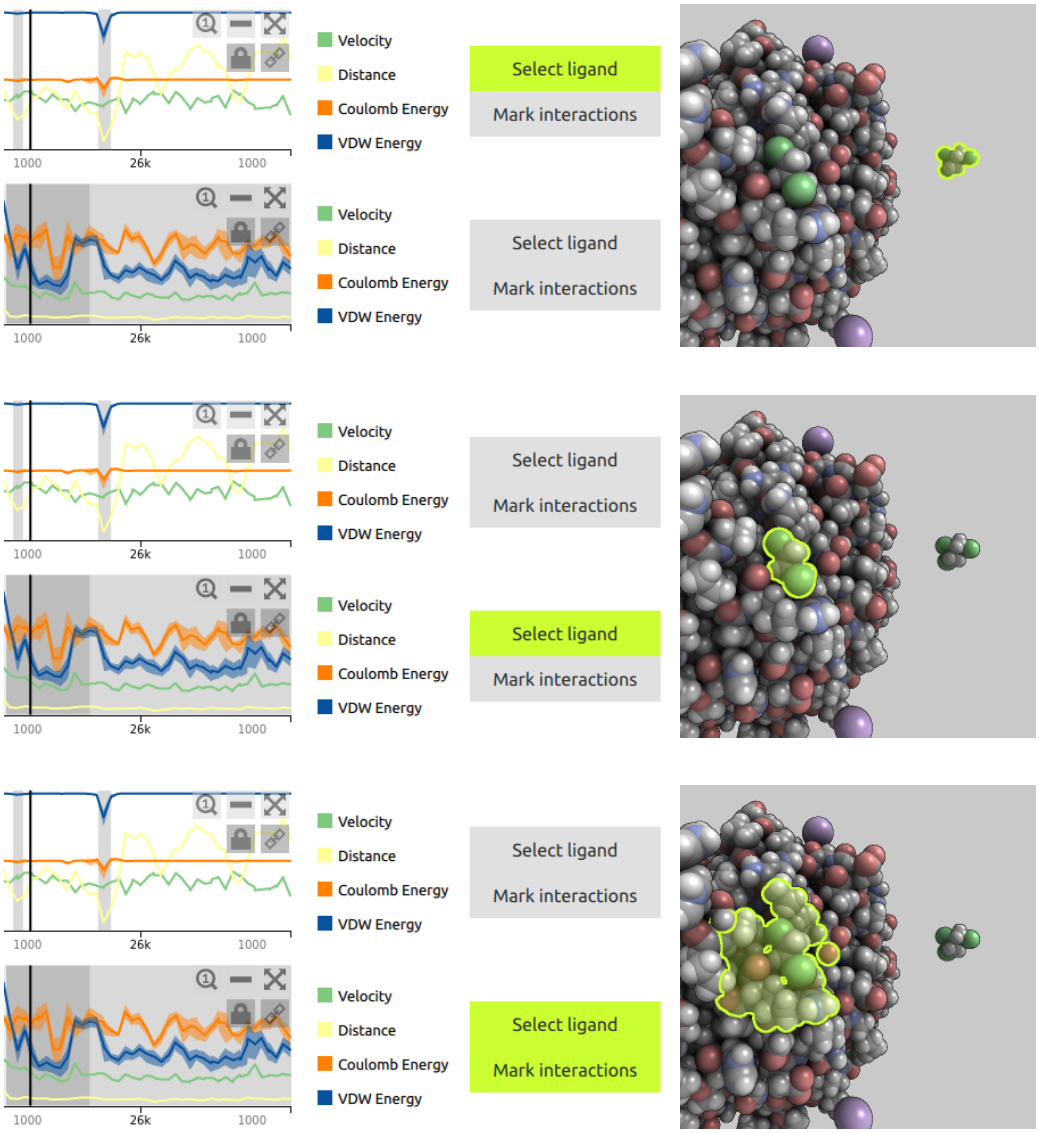


Figure 4.6: The first two rows illustrate how using the buttons the user can identify the 3D representation of the different ligands. Finally, in the third row not only the ligand is highlighted but also all the interacting atoms at the 3D conformation of the current frame. The activated buttons and the atom highlight are of the same color to further strengthen their relationship.

4.3 Chart window

In our visualization system we aim to visualize (at least) four magnitudes (see section 3.2): relative *distance* of the ligand with respect to the protein, ligand's *velocity*

and total *VDW energy* and *Coulomb energy* of each ligand. All these magnitudes are temporally varying, *i. e.* their values change between frames. One of the most widespread representations of such temporally-varying values is using line plots which shows the different values as points in a Cartesian chart, linearly interpolating the in-betweens. However, simply using line plots for visualizing the magnitudes is unreasonable for large trajectories, as the number of points to be shown is far greater than the number of available screen pixels (see Figure 4.7). This is a typical challenge for visualization system of large amounts of data in which one of the most widespread solution is data aggregation. We describe our specific approach to data aggregation using different levels of detail in subsection 4.3.1. Moreover, there are several design considerations when devising a system based on line plots with levels of detail such as how to introduce the perception of the current level of zoom. Such considerations are discussed in subsection 4.3.2.

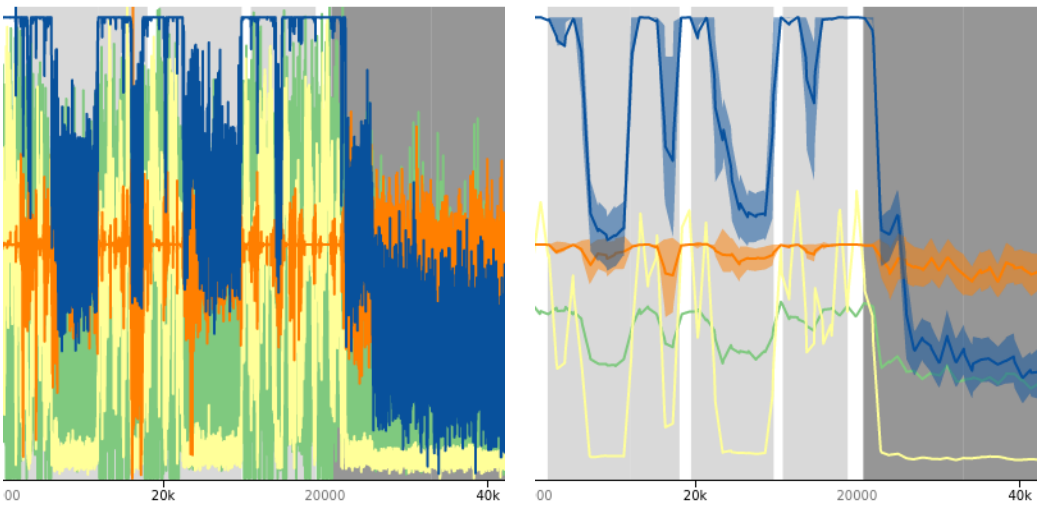


Figure 4.7: Visual comparison between using level-of-detail (right) or plotting the trajectory verbatim (left). The left figure not only suffers from severe visual cluttering but also does not reach interactive frame-rates and is far less informative.

Once decided the visual encoding of the magnitudes, *i. e.* line plots with different levels of detail, another key point to consider is the interaction design. Following the well-known Shneiderman's mantra of overview first, zoom and filter, then details on demand, the charts first illustrates the magnitudes of the whole trajectory in the appropriate level of detail. Then, we implemented the zoom and filter metaphor by allowing the user to decide which interval of frames (viewing window) is to be shown. This way, if the user is

only interested in the details of a handful of frames, those will occupy the whole chart at a finer level of detail. This method allows for a progressive exploration system where the depicted details depend on the current region of interest which the user updates using several interaction techniques explained in subsection 4.3.4.

Since we want to deal with trajectories of many thousands of steps we provide further insight to save the user from spending a lot of time zooming-in and out and dragging back and forth to get the details of the data. We visually encode the derived classification of whether a ligand is *outside* or interacting *inside* or at the *surface* of the protein (see chapter 3) in the background of the chart, as different shades, that can be clicked to zoom into that region. The users can thus quickly direct their attention to the areas where important interactions may be happening. This information is calculated at the highest resolution and hierarchically clustered to be used in zoomed-out views of the chart, so different levels of resolution will have their different clustering as explained later, in subsection 4.3.3. The same clustering strategy is also used when marking the interactions of the ligand with the selected atoms (see subsection 4.2.2).

Finally, we extended the chart system to provide support for multiple ligands. As the magnitudes are relative to each ligand, plotting all four magnitudes per ligand in a single chart does not scale with the number of ligands as with just three ligands there would be already 12 line plots, causing sever visual cluttering. Albeit more intricate visualization systems could be devised, we decided to plot the magnitudes of each ligand in a different coordinated chart. We do so to preserve the simplicity of our visualization and remain consistent with the most common case of a single ligand. Furthermore, all charts are coordinated so that they might be interacted jointly, *i. e.* by sharing the same viewing window, or independently, and they can be hidden in case more screen space is desired for the other charts.

4.3.1 Levels of detail

There are several approaches when implementing data aggregation. For one, multiple representations have been designed to directly illustrate data aggregation such as box plots or heat maps. However, using another representation would require finding appropriate interaction and visualization techniques to link the more abstract representation (*i. e.* the one with data aggregation) with the more detailed one (*i. e.* the original line plots). Instead, for our system, we opted for maintaining the same representation thorough all the levels of data aggregation. We designed a hierarchical representation where each level

is created by a using low-pass filter with a frequency twice as low as its previous level (see Figure 4.8). Finally, using the zoom metaphor, the user may focus on specific regions which progressively acquire more detail as the system internally changes the current used level in the representation hierarchy.

Implementation details

The different levels of detail of the magnitudes are built in a binary fashion: the most fine-grained level data (the original magnitude) are grouped in pairs from which the average, minimum, maximum and first and third quartiles are computed. These pairs are the elements of the second to most fine-grained level. The rest of the levels are built by subsequently grouping in pairs until a single group remains, which is the coarsest level containing the average, minimum, maximum and the first and third quartiles of the whole trajectory.

During the visualization, the transition between levels-of-detail is smoothed out in order to avoid abrupt changes and to improve the spatial cohesion during exploration of the data. This is achieved by a procedure similar to the so-called "tri-linear filtering" in 2D images described as follows. Given a hierarchy of levels-of-detail, the closest two LoD that gives a point for every 5 pixels in the current view are selected, say k_c for the coarser level and k_f for the finer level. Then, a factor $\alpha \in [0, 1)$ is computed which equals to 0 when at level k_f there is exactly a point for every 5 pixels and approaches 1 the closer the level k_c is to have exactly a point for every 5 pixels (incidentally this is equivalent to level k_f having a point for every 2.5 pixels by construction). The points that are finally visualized correspond to the k_f level but instead of their actual value, they take an interpolated value between theirs and the corresponding to the k_c level. For example, imagine that according to k_f level there should be a point at frame 0 with value 6 and one at frame 1 with value 8 and, thus, at level k_c there should be a point at frames 0–1 with value 7 (their average). The resulting plot with an $\alpha = 0.2$ (with linear interpolation) would have a point at frame $0.8 \cdot 0 + 0.2 \cdot 0.5 = 0.1$ with value $0.8 \cdot 6 + 0.2 \cdot 7 = 6.2$ and a point at frame $0.8 \cdot 1 + 0.2 \cdot 0.5 = 0.9$ with value $0.8 \cdot 8 + 0.2 \cdot 7 = 7.8$. The actual interpolation is not linear on α but instead we used a cubic in-out so that the interpolation value tend to be closer to exactly 0 or exactly 1.

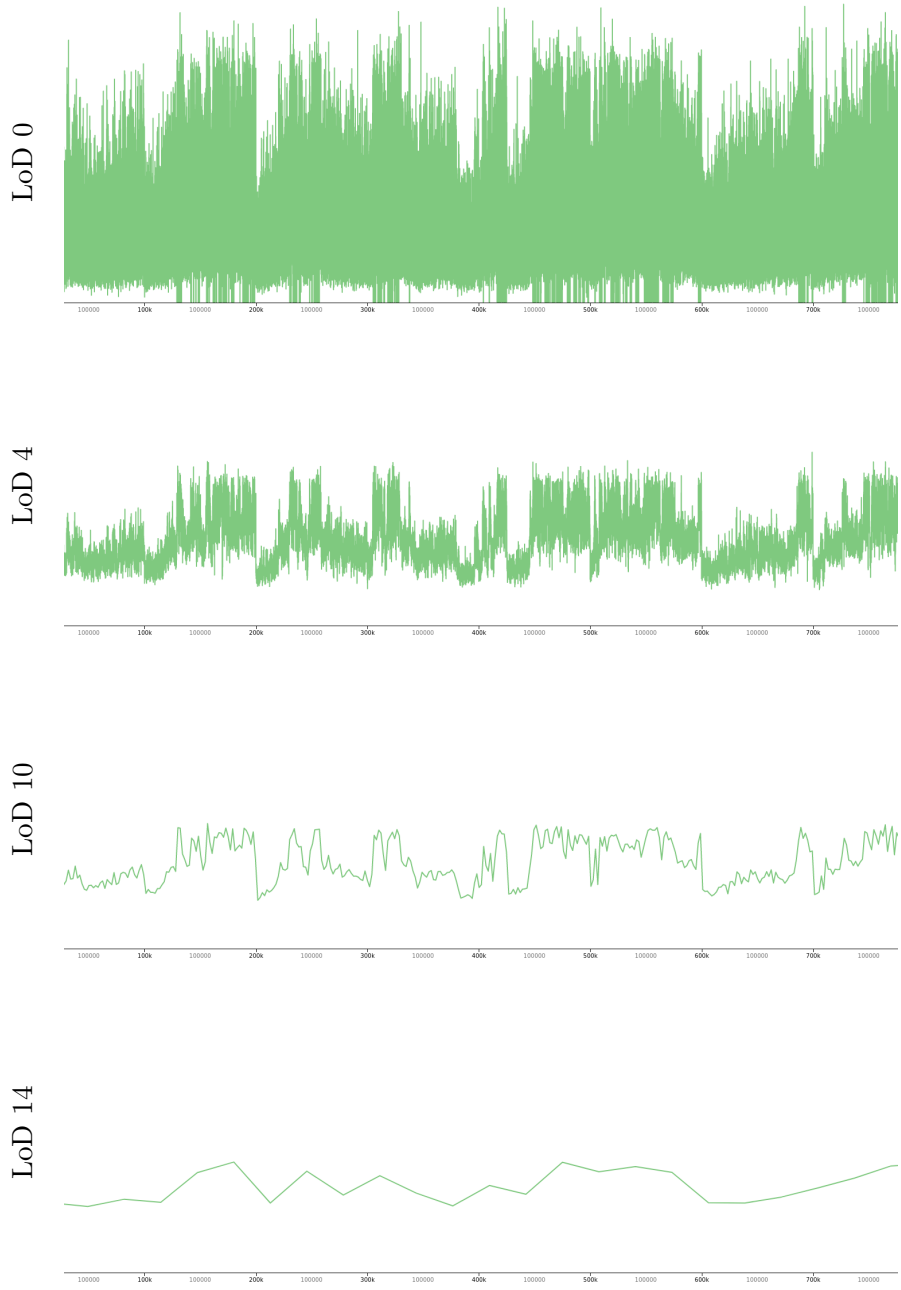


Figure 4.8: Representations of the same values in different levels of detail (LoD). All this plots were obtained by low-pass filters with varying cutoff frequencies. The highest cutoff frequency F_0 used by the level of detail 0 allows all values of the whole trajectory to be depicted, while the cutoff frequency $F_0/2^{14}$ for level 14 only allows a value for every 16 384 frames.

4.3.2 Visual representation details

When designing a visualization system based on 2D plots there are several design considerations. In this section we review the reasonings behind our visualization system characteristics on a variety of topics.

Scale of the y-axis. When plotting multiple heterogeneous magnitudes of different scales and dimensions in a Cartesian chart a problem arise: which is the dimension and scale of the y-axis. For molecular simulations, the absolute value of the magnitudes is not as relevant as which is their tendency. There are several reasons, for one, all the magnitudes come from inexact simulations and computations instead of physical measurements. Moreover, all the magnitudes operate at different scales or even dimensions, greatly undermining the usefulness of a single y-axis scale for all the line plots. Thus, the visualization system linearly normalizes all magnitudes in the $[0, 1]$ domain, while providing the real value as details on demand.

Depicting variance in the data aggregation scheme. Researchers deem the energies as the most important magnitudes. However, as ever less detailed representations are created in the line plot representation hierarchy, the average value of the energies become insufficient for the coarsest level of details. Thus, the system provides insight on the energy variations by displaying the first and third quartile around the average as a shaded region with a less saturated color. The reason why the quartiles are used is because, the magnitudes are normalized, therefore using the maximum and the minimum would result in a much more cluttered and uninformative representation. Moreover, quartiles are more resilient to outliers than the maximum/minimum or the statistical variance while still being widely understood by researchers.

Visualizing context. The differences between viewing an important portion of the trajectory or just a handful of frames are subtle considering that the level of representation changes accordingly. For one, the bottom axis labels change and the user may subtract two consecutive values to get an idea of how many frames are depicted. However, this becomes tedious and error-prone for high values exhibiting low difference such as subtracting 175 075 from 175 150. Thus, we included the number of frames between two axis ticks as a visual cue so that the user may easily evaluate how significant is the viewed

portion of he trajectory. These numbers are shown in a gray color so that they can be easily distinguished from the actual axis labels.

Toggleable legend. For the line plots to be meaningful, a legend of the different line colors must be present. Furthermore, we enhanced the legend by providing an additional use: de-/emphasize the corresponding magnitude. Albeit four line plots usually do not interfere one with each other, there are situations in which one may occlude another. This is specially true if more magnitudes were to be added. In such cases, it may be preferable to hide or de-emphasize one of the magnitudes so the others can be clearly observed. For this purpose, the entries of the legend are buttons that can toggle on/off distance and velocity. For the energy values, the behavior is slightly different: since researchers consider it of utmost importance, when they are on, energy lines are shown and the first and third quartiles around the average are also shown. When toggled off, the line is de-emphasized and the quartile values disappear. Finally, when hovering over the legend entries a tool-tip will appear with their description and units.

4.3.3 Clustering

The hierarchical nature of our data exploration system requires special care when visually encoding per-frame information. One particularly interesting case is when the per-frame information is quantized in a small range of values. Several examples exhibit those characteristics, for example, the derived data of whether a ligand is interacting or not with the protein (the *outside*, *inside*, *surface* classification seen in section 3.2) or a particular set of atoms (the *no interaction*, *low interaction*, *medium interaction*, *high interaction* seen in subsection 4.2.3), but there are others such as whether the energy of a ligand is below a threshold. Therefore, due to its importance, we propose a general system to deal with this type of data in the context of our viewer.

Our main goal is to depict which frames exhibit specific values such as when a ligand is interacting *inside* or at the *surface* of the protein while ignoring the *outside* value. It is worth noting that the number of frames that exhibit such properties may only comprise a minor portion of the overall trajectory. Moreover, we want that the user may easily interact with such regions with, for instance, a simple click. To that end, the size of the displayed regions must be in a balance between a comfortable clickable size while being as accurate and informative as possible. Our proposed solution consists of a hierarchical agglomerative clustering of frame intervals based on the size of the resulting intervals.

This clustering results in a binary tree where the parent interval tightly bounds the intervals of both children, although it may contain empty space if the children were not contiguous.

During the exploration, the system displays the maximal clusters such that no interval is bigger than 80 pixels unless they are terminal, in which case the regions are not artificially split. Given the nature of our hierarchical clustering where the size is penalized, the resulting shown clusters usually exhibit balanced intervals in terms of size. All the resulting intervals are visually encoded as different shades in the background of the chart that can be clicked to zoom and center into that region. Table 4.1 illustrates the color schemes used for the intervals in which the ligand interacts at the *surface* or the *interior* of the protein (see chapter 3), and the intervals in which the ligand interacts with a specific selected atoms (see subsection 4.2.3).

Class	Color single class	Color most prominent class
Interaction		
Interior		
Surface		
Selection		
Low interaction		
Medium interaction		
High interaction		

Table 4.1: Color scheme for the two different clusterings. At the top the color scheme used for the intervals in which the ligand interacts with the protein and below the color scheme used for the intervals in which the interaction happens specifically with the selected atoms. *Color single class* refers to the color used when the interval is composed uniquely with frames of the specified class while *Color most prominent class* is when the most abundant class of the interval is the specified one.

Finally, as more than one clustering might exist in a given time, all the currently possible clusterings are shown in different selectable bars, from which the user may select which clustering should be active and, thus, be interactive and appear in the main chart area as seen in Figure 4.9. As the clustering bars are also color coded the user has accessible at any time the information of all the clusterings and can visually compare them.

Implementation details

Agglomerative hierarchical clustering are built bottom-up, *i. e.* at each step the data points are grouped in ever bigger clusters. For our system, this construction starts with contiguous frames of the same type grouped together in single intervals. For example, given $C_i = [<\text{initial frame}>, <\text{final frame}>, <\text{value}>]$, let the following initial intervals exist at step 0: $C_0 = [0, 4, \textit{interior}]$, $C_1 = [4, 10, \textit{surface}]$, $C_2 = [20, 24, \textit{surface}]$, $C_3 = [30, 58, \textit{interior}]$ and $C_4 = [60, 62, \textit{surface}]$. Then, iteratively the pair of intervals which would result in the smallest-sized valid interval are joined in a cluster forming a new interval that tightly contains both of them. Here valid means there is no overlap with any other interval not included in the cluster. In the previous example, C_0 and C_2 cannot be merged because the resulting interval $C_{0,2} = [0, 24]$ overlaps with $C_1 = [4, 10, \textit{interior}]$ that is not included in $C_{0,2}$. Finally, for each merged interval we consider for its value which is the most prominent type among its components. The previous example would develop as follows:

S0. $C_0 = [0, 4, \textit{interior}]$, $C_1 = [4, 10, \textit{surface}]$, $C_2 = [20, 24, \textit{surface}]$, $C_3 = [30, 58, \textit{interior}]$, $C_4 = [60, 62, \textit{surface}]$.

S1. Merge C_0 with C_1: $C_{0,1} = [0, 10, \textit{mostly surface}]$, $C_2 = [20, 24, \textit{surface}]$, $C_3 = [30, 58, \textit{interior}]$, $C_4 = [60, 62, \textit{surface}]$.

S2. Merge $C_{0,1}$ with C_2: $C_{0,1,2} = [0, 24, \textit{mostly surface}]$, $C_3 = [30, 58, \textit{interior}]$, $C_4 = [60, 62, \textit{surface}]$.

S3. Merge C_3 with C_4: $C_{0,1,2} = [0, 24, \textit{mostly surface}]$, $C_{3,4} = [30, 62, \textit{mostly interior}]$.

S4. Merge $C_{0,1,2}$ with $C_{3,4}$: $C_{0,1,2,3,4} = [0, 62, \textit{mostly interior}]$.

The construction of such a hierarchical clustering with n initial intervals can be achieved in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space with the use of a heap. This contrasts with the usual $\mathcal{O}(n^2)$ time complexity of algorithms such as SLINK [31] (for single-linkage) or CLINK [8] (for complete-linkage) since, in our case, the number of possible pairs is linear instead of quadratic.

4.3.4 Chart interaction

All the chart interactions can be divided according to their purpose into three different classes: interactions that affect the coordination between charts, that modify

the current viewing window (*i. e.* the interval of frames currently visualized) or that provide more details. Figure 4.9 illustrate the different parts of the chart view. For this section, we strongly encourage the reader to watch the provided supplemental video.

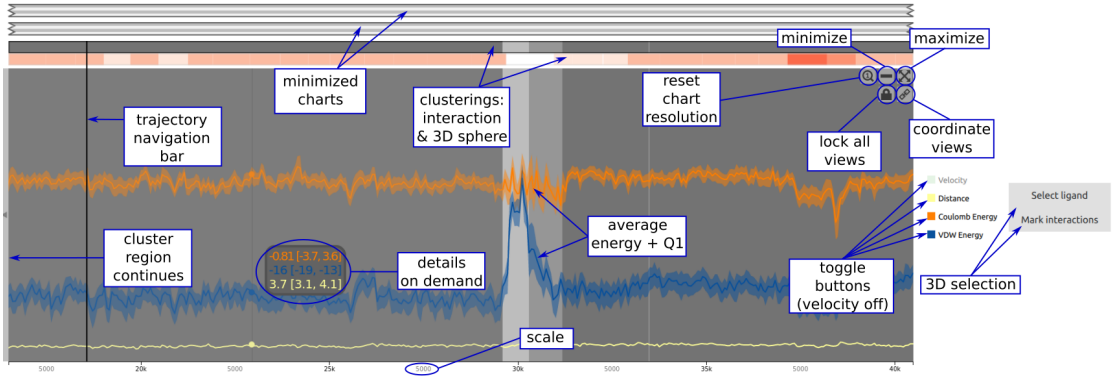


Figure 4.9: All the visual and interactive elements of the chart. The user can get the details-on-demand by hovering over the chart with the `Ctrl` button pressed. The charts can be minimized to give more room for other charts, and an icon simulating a folded paper provides a visual cue on the existence of other (minimized) charts.

All the interactions that affect coordination between charts are done activating four of the five buttons of the top right corner:

Minimize/maximize The charts can be individually minimized, to provide more room for the other charts, by clicking the minus button. The maximization button minimizes the other charts.

Lock all charts All charts are set to the same zooming level and position as the one we have clicked on. By locking, further chart exploration (with any chart) is coordinated.

Coordinate charts The link icon button sets the other charts to the same zoom level and position as the current one, but the coordination is not locked.

The reason why these interaction techniques are implemented as buttons is due to its relatively infrequent usage combined with the fact that they are complex actions with some of them being historically implemented as buttons, *e. g.* minimize and maximize.

Most of these interactions are direct manipulations due to its primordial importance. The only exception is one of the top-right buttons, the one containing the magnifying lens icon with a number one inside, that switches back to the initial size and position of charts. The rest of manipulations are

Zoom in a cluster The user may zoom in by clicking on the cluster region, which will magnify the selected region. Right-click will zoom back to the previous zooming level and position. Hovering over the chart highlights the cluster that would be selected with a left click.

Continuous zooming-in/out With the mouse wheel, the user can achieve detailed zooming.

Dragging The user can drag around the chart with the left mouse button.

Center cluster If a selected region (in a cluster) does not fit in the current view, arrows at the left or right part of the chart indicate that the region continues outside the chart. These arrows act as buttons that can be clicked to center the cluster on screen.

Explore range Dragging with the right button lets the user define an arbitrary range, which is then magnified to fill the chart.

All these actions either have an immediate effect, *e. g.* continuous zooming-in/out, or provide visual feedback on what will be the effect, *e. g.* the brush motif when exploring a range or the highlight of the cluster that will be zoomed/centered. They provide means for the user to explore the data and, thus, due to their relevance, they are triggered by the primary mouse buttons and the wheel.

Finally, there are basically two direct manipulations that provide more details:

Details on demand Control button opens the detail view, providing data on the actual values of the charts as the user hovers over them.

Set the navigation point By `Ctrl`+click in any chart, the user can set the navigation step to be displayed in the 3D viewport.

The interaction is consistent in that Control button refers to the details. However, both types of provided details are fundamentally different. Whereas *details on demand* refer to providing what are the actual values of the line plots, setting the navigation point is how the 3D view is updated so that the user may explore the molecular system conformation at

a given point in time. Albeit the importance of the 2D to 3D link, all the primary mouse buttons are already used by the different data exploration interactions. Thus, in order to preserve consistency among the interactions, the 2D to 3D have been implemented as `Ctrl`+click.

4.4 Other interface design considerations

When designing an integrated system there are several considerations and general guidelines that do not entirely fall into either of its components. The following list include some of the guidelines we followed in order to provide a better user experience:

- All the components are aligned and the overall interface is balanced.
- The colors were selected among colors that contrasted well using the Color Brewer system.
- We followed the Shneiderman's mantra of overview first, zoom and filter, then details on demand, when applicable, *e. g.* when designing the different interaction and visualization techniques for progressive data exploration.
- When possible, we leveraged the user prior knowledge, reducing the required cognitive effort and, thus, lowering the learning curve of the system, *e. g.* the iconography of minimizing/maximizing or the use of line plots as an abstraction representation of the molecular simulation trajectory.
- The different components are semantically organized, strengthening the link between related actions, *e. g.* the highlight the ligand and highlight the contact atoms buttons next to the corresponding chart of the ligand.

5 Results

The presented visualization system starts with a general view and greatly facilitates low-level analysis by providing several data-derived cues that guide the user to the regions of interest. This low-level analysis is indeed a necessity since the overview lacks enough detail for large trajectories that amount to hundreds of thousands of steps. In this chapter we provide two use cases of our system that have been validated by an expert. Finally, we also provide details on the overall performance and discuss the advantages and limitations of the presented system. All the times and performance measures were taken in a computer with an Intel(R) Core(TM) i7-6700 CPU (3.40 GHz), 32 GB of RAM and a NVIDIA GeForce GTX 660 Ti.

5.1 Use cases

In the following, we provide some examples of use of our application.

Classical 2D to 3D exploration. Molecular dynamics trajectory analysis typically starts with the researcher examining the energy plots in order to gain a high-level understanding of the whole simulation. However, this initial analysis may require a lot of manual labor for large trajectories as simple plots become unfeasible and any non-interactive general view will lack the necessary details for a throughout evaluation. This leads to the situation where the researcher may be forced to break the trajectory into small amounts of steps evaluated independently, which is a tedious and laborious work without the proper tools.

The process of exploring the energy plots is greatly accelerated with our software as it shows thousands of steps at a glance in a 2D plot, and the zooming-in tools facilitate drilling down to the details quickly, while providing convenient ways to return to the initial overview. Once the researcher requires viewing the atom conformation of a specific part of the trajectory, they may update the 3D view simply by `Ctrl`+click

in corresponding the 2D plot position. At this point, the user may explore the whole 3D scene using different motifs, from the van der Waals surface based representation to the solvent excluded surface representation, and the culling plane tool combined with a semitransparent silhouette to explore the interior of the protein while preserving the overall form.

Finally, each ligand has a corresponding charts that can be inspected individually, if there are portions of the trajectory that are interesting for the different ligands, or jointly (*i. e.* coordinating the charts), if the user wants to globally analyze certain steps of the trajectory.

Exploratory 3D to 2D. A novel technique of our system is the link between the 3D view and the 2D charts. Researchers usually have a priori knowledge of the protein binding sites and might be interested in querying whether a specific site has interacted with the ligand(s) during the simulation. Without the appropriate tool, the user would have to examine all the portions of the trajectory that exhibit the desired characteristics, *e. g.* low energy indicating stable interactions, and explore the 3D atom configuration in order to determine whether the interaction happened on the expected active site. Instead, using our system, the user may select the 3D region of interest and the steps where the different ligand(s) interact with the selected atoms will be highlighted. Moreover, exploration of the new highlighted regions is greatly facilitated by the adaptive zooming-in tools of the system. Therefore, this method not only answers the question of whether a binding site have been visited, but also at which trajectory intervals do the ligands interact while providing exploration tools for a more detailed close-up.

5.2 Evaluation

To obtain expert feedback, we have exposed our application to a molecular scientist working with molecular simulation data on a regular basis. He emphasized that he likes this new way of investigating trajectories, whereby he pointed out the following things which he likes the most: *i)* the interface, *ii)* the utility, his opinion is that the visualization system seems very useful to analyze the interactions and the behavior of ligands, and *iii)* the representation of the contacting atoms.

The domain expert also gave several suggestions to further improve the proposed system. He, for instance, proposed to use different representations for residues exhibiting stronger van der Waals and electrostatic interactions. Unfortunately, the input data we

had did not provide this fine grain information. Should this be the case, the representation seems not to be problematic, but the amount of information to process might be huge.

5.3 System performance

Our software is able to load any of the datasets mentioned in chapter 3 in mere seconds. This is possible due to the *lazy loading* approach in which the system maintains most of the data out-of-core and initialize all the necessary structures so that only the needed data is fetched in a on-demand basis, *e. g.* file handlers with an associated *seek* pointer and frame stride. Thus, the total loading time is comprised by the relatively small topology file parse and the magnitudes load and transfer time to the chart viewer in which the hierarchical level-of-detail and clustering structure are built, besides the loading time of a single 3D configuration corresponding to the first trajectory frame. None of these tasks scale to the number of frames except the magnitudes load and transfer time and the corresponding initialized data structures. However, note that the several magnitudes amount to a minor portion of the total data as hundreds of thousands of single-precision 32-bit floats are about $\sim$0.5 MB each magnitude, which is an easily manageable size for today's technology. It is worth mentioning that this does not include the preprocessing time, which is about 34 s and 7 min 56 s for the 800 K frames trajectory and the 50 K frames with three ligands, respectively.

Even with the *lazy loading* in which each frame is loaded in a on-demand basis, we reach real-time frame-rates (above 30 fps) during animation for most of the 3D representations except for the non-progressive solvent excluded surface representation. In contrast, the system achieves real-time frame-rates in the 3D interaction of a single frame as the most expensive step in the non-progressive solvent excluded surface representation is at the initialization of the frame, *i. e.* when computing the corresponding geometry. The web engine is also able to render in real-time the generated interactive SVG plots thanks to the combination of both the use of level-of-details and the data window we maintain in order to avoid plotting superfluous points that lie outside the current view.

Finally, all the implemented tools run in interactive times including the 3D selection with the posterior hierarchical clustering (see subsection 4.3.3) and the highlight of the interacting atoms in the current frame with the selected ligand (see subsection 4.2.2).

5.4 Discussion

The major benefits of our solution with respect to the other approaches for molecular dynamic trajectory exploration are the following:

- The system is able to cope with large trajectories that amount to hundreds of thousands of steps. We have demonstrated this with molecular simulations of up to 800 K steps but larger trajectories are definitely possible at the expense of a longer pre-process time.
- The interactive enhanced charts allow for a progressive inspection of large-scale molecular dynamics simulation in an abstraction commonly used by the researchers, *i. e.* linear plots of several magnitudes including energies, while exposing more details in an intuitive and seamless way by means of the zoom metaphor. Moreover, the exploration of regions of interest derived from the data is greatly facilitated by using actionable widgets that adapt to the current level of zoom.
- The analysis of multiple ligands at once is supported by means of using different coordinated charts, which can be interacted either jointly or independently. This contrast with most of the available software were only a single ligand can be considered at a time, disregarding interesting combined effects and their analysis.
- The novel bidirectional link between the 2D and 3D provides a new way to interact with molecules. Most existing visualization approaches, even commercial ones, separate the exploration of the 2D energy charts from the 3D view. In contrast, our system nicely integrates both views allowing the user to quickly jump from the charts to the specific corresponding 3D conformation, which is necessary for a sound analysis of the simulation. Furthermore, the system integrates the reverse workflow in which the researcher may select specific active sites in the 3D view in order to explore in which frames the ligand(s) interact with the selected region and how this is reflected in the different magnitudes.

On the contrary, there are also some limitations regarding our system, specifically:

- As previously seen, the current chart visualization can handle up to three ligands at once. However, it would become too cluttered if more than three need to be shown simultaneously without temporarily hiding the 3D view. Using minimization this issue is mitigated but not completely erased as the minimized charts also take up

some screen space. Therefore, we expect that the system is capable of holding up to 7-8 minimized charts before it became too cluttered. Nevertheless, most of the molecular simulations deals with a single ligand in which cases this is a non-issue.

- Our system currently only offers two kind of clustering strategies: the regions of interest based on derived data and the regions where the ligand interacts with the 3D selection. However, our clustering approach is general enough to support several other types of classifications, such as whether the energy values are below a certain threshold. This may become an issue for our current visualization of the different available clusterings, *i. e.* the top bars of the chart, as it is not scalable beyond a handful of clusterings. This is specially aggravated by the fact that each bar is present for each ligand in their corresponding charts, thus furtherer reducing the available space for the magnitudes plots.

- The proposed *lazy loading* reaches real-time frame-rate during animation due to the relative small amount of simulated atoms (about 35 K atoms in the cases with explicit solvent positions for our dataset). For real-time animation of bigger proteins the amount of data needed to be read per frame would require more complex solutions such as dividing the trajectory into chunks that are read in a secondary thread according to a predictor.

6 Conclusions

New technological and algorithmic advances in molecular simulations require novel visualization and exploration techniques in order to deal with the increasingly huge amount of generated data. Such techniques, not only must manage the volume of the data itself, but must provide the user with the necessary tools for an efficient exploration and analysis so that new insights are easily acquired without the need of laborious and tedious work. Most current approaches fail in providing an integrated system in which both, the 3D conformation details of the simulation and an abstraction view that allow for a progressive exploration of the whole trajectory, work cohesively in facilitating the acquisition of important high-level knowledge such as when, which and how the different ligand(s) interact with specific active sites of the protein, or at which intervals the molecular system undergo critical interactions that must be thoroughly examined. Hence, in this work we present a visualization system that overcome all the aforementioned shortcomings and which we strongly believe will be of great use in analyzing the interactions and the behavior of ligands in large molecular dynamics trajectories, belief that has been already supported by a domain expert.

Regarding the devision and implementation of our system we carefully considered several approaches and we were guided by the principles of creating a user-friendly system that required the minimum cognitive effort for the user. Given the ubiquity of line plots in the researchers work, they were a sensible choice for a progressive exploration system of the whole trajectory, specially considering how biochemists already use energy plots for analyzing how the simulations developed. Thus, combined with data-driven cues and several well-known chart interactions and metaphors, such as zoom, pan and brush, the resulting system is both intuitive and informative for the user. Moreover, the use of the 2D chart has been extended to multiple ligands by plotting each magnitude to the corresponding separate chart of the ligand. All these charts are coordinated and allow for an individual or joint analysis of all the ligands at once. Finally, we aimed for an integrated system with the 3D view of the atom configuration as this is fundamental for understanding the molecular processes. Combined with the novel interaction techniques

in which both the 3D view and the 2D charts are linked, the researcher can quickly shed light upon crucial inquiries that would otherwise be cumbersome or impossible to answer in a non-integrated system, *e. g.* when do the ligands interact with specific active sites or which 3D conformation correspond to intervals characterized by certain values of, for instance, the energy.

6.1 Future work

Various potential extensions and future lines of research exist. For one, were the dynamic addition and removal of single frames be desirable features, they could be reintroduced by adapting the calculation of the different magnitudes to an incremental procedure. Other possible extensions include facilitating the analysis of several trajectories at once or to provide a more detailed inspection of the protein active sites. Albeit our modular system could be modified to include any of these features, none is straightforward as there are visual and interaction implications to consider, *e. g.* how to visually encode several 3D conformations of different trajectories or which characteristics are the most important to depict of the active sites according to the target use cases.

Finally, we plan to perform a broader evaluation with more researchers involved to ask them their opinion of our system and assess the desirability of several ideas for future features. This way, we can jointly evaluate which novel visualization tools and methods are required for improving their daily work which will positively impact in their research.

6.2 Publication

Due to its novelty and already mentioned benefits, the work presented in this thesis have been submitted to the 2018 IEEE Scientific Visualization (SciVis) conference under the following entry:

- David Duran, Pedro Hermosilla, Timo Ropinski, Barbora Kozlíková, Àlvar Vinacua, Pere-Pau Vázquez. “Visualization of Large Molecular Trajectories”. Submitted to: *IEEE Scientific Visualization* (2018).

References

[1] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3 Data-Driven Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (Dec. 2011), pp. 2301–2309. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.185. URL: http://dx.doi.org/10.1109/TVCG.2011.185.

[2] Jan Byska, Adam Jurcik, Meister Eduard Gröller, Ivan Viola, and Barbora Kozlikova. "MoleCollar and Tunnel Heat Map Visualizations for Conveying Spatio-Temporo-Chemical Properties Across and Along Protein Voids". In: *Computer Graphics Forum* 3.34 (May 2015). EuroVis 2015 - Conference Proceedings, pp. 1–10. URL: https://www.cg.tuwien.ac.at/research/publications/2015/Viola_Ivan_2015_MCT/.

[3] Jan Byška, Mathieu Le Muzic, M Eduard Gröller, Ivan Viola, and Barbora Kozlikova. "AnimoAminoMiner: Exploration of Protein Tunnels and their Properties in Molecular Dynamics". In: *IEEE transactions on visualization and computer graphics* 22.1 (2016), pp. 747–756.

[4] David A Case, Thomas E Cheatham, Tom Darden, Holger Gohlke, Ray Luo, Kenneth M Merz, Alexey Onufriev, Carlos Simmerling, Bing Wang, and Robert J Woods. "The Amber biomolecular simulation programs". In: *Journal of computational chemistry* 26.16 (2005), pp. 1668–1688.

[5] Matthieu Chavent, Bruno Lévy, Michael Krone, Katrin Bidmon, Jean-Philippe Nominé, Thomas Ertl, and Marc Baaden. "GPU-powered tools boost molecular visualization". In: *Briefings in Bioinformatics* 12.6 (2011), pp. 689–701.

[6] Wei Chen, Fangzhou Guo, and Fei-Yue Wang. "A survey of traffic data visualization". In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 2970–2984.

[7] Erick Cuenca, Arnaud Sallaberry, Florence Ying Wang, and Pascal Poncelet. "MultiStream: A Multiresolution Streamgraph Approach to Explore Hierarchical Time Series". In: *IEEE Transactions on Visualization and Computer Graphics* (2018).

[8] Daniel Defays. "An efficient algorithm for a complete link method". In: *The Computer Journal* 20.4 (1977), pp. 364–366.

[9] Fan Du, Ben Shneiderman, Catherine Plaisant, Sana Malik, and Adam Perer. "Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus". In: *IEEE transactions on visualization and computer graphics* 23.6 (2017), pp. 1636–1649.

[10] Katarína Furmanová, Miroslava Jarešová, Jan Byška, Adam Jurčík, Július Parulek, Helwig Hauser, and Barbora Kozlíková. "Interactive exploration of ligand transportation through protein tunnels". In: *BMC bioinformatics* 18.2 (2017), p. 22.

[11] Parke Godfrey, Jarek Gryz, and Piotr Lasek. "Interactive visualization of large data sets". In: *IEEE Transactions on Knowledge and Data Engineering* 28.8 (2016), pp. 2142–2157.

[12] S. Grottel, M. Krone, C. Muller, G. Reina, and T. Ertl. "MegaMol – A Prototyping Framework for Particle-based Visualization". In: *Visualization and Computer Graphics, IEEE Transactions on* 21.2 (Feb. 2015), pp. 201–214. ISSN: 1077-2626.

[13] Pedro Hermosilla. "Advanced inspection techniques for molecular simulations". Under the joint supervision of Pere Pau Vázquez and Àlvar Vinacua. PhD thesis. Universitat Politècnica de Catalunya, 2017.

[14] Pedro Hermosilla, Jorge Estrada, Victor Guallar, Timo Ropinski, Àlvar Vinacua, and Pere-Pau Vázquez. "Physics-Based Visual Characterization of Molecular Interaction Forces". In: *IEEE transactions on visualization and computer graphics* 23.1 (2017), pp. 731–740.

[15] David C Hoaglin. "John W. Tukey and data analysis". In: *Statistical Science* (2003), pp. 311–318.

[16] William Humphrey, Andrew Dalke, and Klaus Schulten. "VMD: visual molecular dynamics". In: *Journal of molecular graphics* 14.1 (1996), pp. 33–38.

[17] Wolfgang Kabsch and Christian Sander. "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features". In: *Biopolymers* 22.12 (1983), pp. 2577–2637.

[18] Barbora Kozlíková, Michael Krone, Martin Falk, Norbert Lindow, Marc Baaden, Daniel Baum, Ivan Viola, Julius Parulek, and H-C Hege. "Visualization of biomolecular structures: State of the art revisited". In: *Computer Graphics Forum*. Vol. 36. 8. Wiley Online Library. 2017, pp. 178–204.

[19] Michael Krone, Barbora Kozlikova, Norbert Lindow, Marc Baaden, Daniel Baum, Julius Parulek, H-C Hege, and Ivan Viola. "Visual analysis of biomolecular cavities: State of the art". In: *Computer Graphics Forum*. Vol. 35. 3. Wiley Online Library. 2016, pp. 527–551.

[20] Roman A Laskowski and Mark B Swindells. *LigPlot+: multiple ligand–protein interaction diagrams for drug discovery*. 2011.

[21] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. "cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets." In: *VCBM*. 2015, pp. 61–70.

[22] IV Likhachev, NK Balabaev, and OV Galzitskaya. "Available instruments for analyzing molecular dynamics trajectories". In: *The open biochemistry journal* 10 (2016), p. 1.

[23] Norbert Lindow. "Dynamic Channels in Biomolecular Systems: Path Analysis and Visualization". In: *Proceedings of the 2012 IEEE Symposium on Biological Data Visualization (BioVis)*. BIOVIS '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 99–106. ISBN: 978-1-4673-4729-7. DOI: 10.1109/BioVis.2012.6378599. URL: http://dx.doi.org/10.1109/BioVis.2012.6378599.

[24] Zhicheng Liu, Bernard Kerr, Mira Dontcheva, Justin Grover, Matthew Hoffman, and Alan Wilson. "CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences". In: *Computer Graphics Forum*. Vol. 36. 3. Wiley Online Library. 2017, pp. 527–538.

[25] Zhicheng Liu, Yang Wang, Mira Dontcheva, Matthew Hoffman, Seth Walker, and Alan Wilson. "Patterns and sequences: Interactive exploration of clickstreams to understand common visitor paths". In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 321–330.

[26] Maestro, Schrödinger, LLC. *Schrödinger Release 2018-1: Desmond Molecular Dynamics System*. Maestro-Desmond Interoperability Tools, Schrödinger. D. E. Shaw Research, New York, NY, 2018.

[27] Sana Malik, Ben Shneiderman, Fan Du, Catherine Plaisant, and Margret Bjarnadottir. "High-volume hypothesis testing: Systematic exploration of event sequence comparisons". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 6.1 (2016), p. 9.

[28] Daniel R Roe and Thomas E Cheatham III. "PTRAJ and CPPTRAJ: software for processing and analysis of molecular dynamics trajectory data". In: *Journal of chemical theory and computation* 9.7 (2013), pp. 3084–3095.

[29] Sebastian Salentin, Sven Schreiber, V Joachim Haupt, Melissa F Adasme, and Michael Schroeder. "PLIP: fully automated protein–ligand interaction profiler". In: *Nucleic acids research* 43.W1 (2015), W443–W447.

[30] Daniel Seeliger and Bert L de Groot. "Ligand docking and binding site analysis with PyMOL and Autodock/Vina". In: *Journal of computer-aided molecular design* 24.5 (2010), pp. 417–422.

[31] Robin Sibson. "SLINK: an optimally efficient algorithm for the single-link cluster method". In: *The computer journal* 16.1 (1973), pp. 30–34.

[32] Charles D Stolper, Adam Perer, and David Gotz. "Progressive visual analytics: User-driven visual exploration of in-progress analytics". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1653–1662.

[33] Max C Watson. "Time maps: A tool for visualizing many discrete events across multiple timescales". In: *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE. 2015, pp. 793–800.

[34] Krist Wongsuphasawat and David Gotz. "Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2659–2668.

[35] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. "LifeFlow: visualizing an overview of event sequences". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2011, pp. 1747–1756.

[36] Jian Zhao, Zhicheng Liu, Mira Dontcheva, Aaron Hertzmann, and Alan Wilson. "MatrixWave: Visual comparison of event sequence data". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 259–268.